OBT1553-IP 核

用户手册

欧比特控制工程股份有限公司

2008 年 10 月

目 录

目 录.....3

表目录.....5

图目录.....6

1. 简介.....7

1.1 概要说明.....7
1.2 主要特征.....7
1.3 结构框图.....9

2. 主要功能模块概述.....12

2.1 总线控制器（BC）.....12
2.2 远程终端（RT）.....12
2.3 总线监视器（BM）.....13

3. 寄存器描述.....13

3.1 寄存器地址分配表.....13
3.2 中断屏蔽寄存器（IMR）.....14
3.3 BC配置寄存器 1(BC-CFG1).....15
3.4 RT配置寄存器 1 (RT-CFG1).....15
3.5 配置寄存器 2 (CFG2).....16
3.6 启动/复位寄存器（SRR).....16
3.7 BC/RT/BM命令堆栈指针寄存器（STACK_ADDR).....17
3.8 BM初始命令堆栈指针寄存器（INIT_STACK_ADDR).....17
3.9 时间标签寄存器 0（TTR0）.....17
3.10 中断状态寄存器（INT_STA）.....17
3.11 配置寄存器 3（CFG3）.....18
3.12 配置寄存器 4(CFG4).....18
3.13 配置寄存器 5 (CFG5).....19
3.14 BM数据堆栈指针寄存器（BM_STACK_ADDR).....19
3.15 1MBPS/10MBPS配置寄存器（1M_10M_SEL).....19
3.16 BC帧时间/RT上一命令字寄存器(LAST_CMD).....20
3.17 RT状态字寄存器(RT_STA).....20
3.18 RT BIT字寄存器(RT_BIT_REG).....20
3.19 时间标签寄存器 1（TTR1）.....21
3.20 时间标签寄存器 2（TTR2）.....21
3.21 空消息寄存器（IDLE_REG）.....21
3.22 BC控制字（BC_CTRL）.....22
3.23 BC命令字（BC_CMD）.....22
3.24 BC块状态字(BC_BLK).....23
3.25 RT子地址控制字(RT_SUB_CTRL).....23
3.26 RT/BM块状态字(RT/BM_BLK).....24

4. 功能模块描述.....25

4.1 BC总线控制器工作方式.....25

4.1.1 BC存储器地址分配.....25

4.1.2 BC存储器管理.....26

4.1.3 BC消息格式.....26

4.2 RT远程终端工作方式.....27

4.2.1 RT存储器地址分配.....27

4.2.2 RT存储器查找表.....28

4.2.3 IP核实现的方式代码.....29

4.2.4 RT单缓冲存储器管理.....30

4.2.5 RT循环缓冲存储器管理.....30

4.2.6 RT双缓冲存储器管理.....31

4.3 BM总线监视器工作方式.....31

4.3.1 BM存储器地址分配.....31

4.3.2 BM存储器管理.....32

4.3.3 BM子地址选择设置区地址分配.....32

5. 时序图.....33

6. 应用案例.....34

6.1 1MBPS外围接口.....34

6.2 10MBPS外围接口.....35

6.3 BC总线控制器应用案例.....35

6.4 RT远程终端应用案例.....36

6.5 BM总线监视器应用案例.....37

7. 附录.....39

7.1 附录一：BC示例程序.....39

7.2 附录二：RT示例程序.....43

7.3 附录三：BM示例程序.....47

8. 参考资料.....52

表目录

表 1-1 OBT1553 IP 核各模块说明 9
表 1-2 OBT1553B IP 核外部接口信号说明 10
表 3-1 寄存器地址分配 13
表 3-2 中断屏蔽寄存器（IMR） 14
表 3-3 BC配置寄存器 1(BC-CFG1) 15
表 3-4 RT配置寄存器 1(RT- CFG1) 15
表 3-5 配置寄存器 2(CFG2) 16
表 3-6 启动/复位寄存器（SRR） 16
表 3-7 BC/RT命令堆栈指针寄存器(STACK_ADDR) 17
表 3-8 BM初始命令堆栈指针寄存器（INIT_STACK_ADDR) 17
表 3-9 时间标签寄存器 0（TTR） 17
表 3-10 中断状态寄存器（INT_STA） 17
表 3-11 配置寄存器 3(CFG3) 18
表 3-12 配置寄存器 4(CFG4) 18
表 3-13 配置寄存器 5(CFG5) 19
表 3-14 BM数据堆栈指针寄存器（BM_STACK_ADDR) 19
表 3-15 1MBPS/10MBPS配置寄存器（1M_10M_SEL) 19
表 3-16 BC帧时间/RT上一命令字寄存器（LAST_CMD) 20
表 3-17 RT状态字寄存器(RT_STA) 20
表 3-18 RT BIT字寄存器(RT_BIT_REG) 20
表 3-19 时间标签寄存器 1（TTR1） 21
表 3-20 时间标签寄存器 2（TTR2） 21
表 3-21 空消息寄存器（IDLE_REG） 21
表 3-22 BC控制字（BC_CTRL） 22
表 3-23 BC命令字(BC_CMD) 22
表 3-24 BC块状态字(BC_BLK) 23
表 3-25 RT子地址控制字(RT_SUB_CTRL) 23
表 3-26 RT/BM 块状态字(RT/BM_BLK) 24
表 4-1 BC存储器地址分配(4K双口RAM) 25
表 4-2 BC消息格式 26
表 4-3 BC消息格式（接上表） 27
表 4-4 RT存储器地址分配(4K 双口RAM) 27
表 4-5 RT存储器查找表(LOOK_UP TABLE) 28
表 4-6 IP核实现的方式代码 29
表 4-7 BM存储器地址分配 31

图目录

图 1-1 OBT1553B IP 核结构框图 9
图 4-1 BC存储器管理 26
图 4-2 RT单缓冲存储器管理 30
图 4-3 RT循环缓冲存储器管理 30
图 4-4 RT 双缓冲存储器管理 31
图 4-5 BM存储器管理 32
图 5-1 发送波形 33
图 5-2 接收波形 33
图 5-3 数据频率编码之间的关系 33
图 6-1 OBT1553-1M外围接口原理图 34
图 6-2 OBT1553-10M接口原理图 35
图 6-3 OBT1553 BC程序流程图 36
图 6-4 OBT1553 RT程序流程图 37
图 6-5 OBT1553 BM程序流程图 38

1. 简介

1.1 概要说明

OBT1553 IP 核是依据 1553B 总线协议(1553A/B Notice2 协议)和参考 DDC-ACE 系列芯片（主要是 61580） 设计的 IP 核，能方便地挂载到 AMBA APB2.0 总线上。 1553 IP 核主要包括通信协议模块 A、通信协议模块 B、主机信号接口模块、配置寄存器模块、存储管理模块和时钟管理与主控制模块等。其中 1553 通信协议模块部分用差分曼彻斯特编码实现时分命令响应式串行通讯，主要包括 A、B 通道编码器和解码器；主机信号接口模块实现 1553 IP 核与 CPU 的接口,用来实现 CPU 对 OBT1553 IP 核的控制;配置寄存器主要实现对 OBT1553 IP 核功能的配置，能够间接反映 1553 IP 核的功能；存储管理为 CPU 和 1553 IP 核之间交互数据的管理方式，主要为 4K*16BIT 的双口 RAM； 时钟管理与主控制模块实现对输入时钟的管理和对 OBT1553 IP 核的基本通讯功能的实现与控制，是该 IP 核的中心控制单元能配置成 BC、RT、BM 三种类型的控制器。同时该 IP 核还支持 10Mbps 的总线传输速度。

OBT1553 IP 核主要应用在航空、航天测控网络、工业控制等技术领域。

1.2 主要特征

- 通过硬件逻辑方式完全实现 MIL-STD-1553B 标准(国军标 GJB289A-97 标准);
- 操作方式、寄存器设置以及存储器布局等方面同 BU-61580 兼容；
- 支持的通讯类型包括：
  - BC → RT；
  - RT → BC；
  - RT → RT；
  - Broadcast；
  - Mode code；
- 能被配置为 BC、RT、BM 三种类型的控制器；
- 支持 1Mbps、10Mbps 两种传输速度；
- 带 4K*16Bit 的集成 DPRAM，根据需要可进行集成 DPRAM 的裁减；
- 具有同步和异步两种形式的 HOST 接口：
  - 同步 HOST 接口：符合 AMBA2.0 总线标准;；
  - 异步 HOST 接口：支持 INTEL 和 MOTOROLA 两种时序模式；
- 外部接口支持通用的 1553B 总线收发器：HI1567、HI1573 等
- 带 A、B 双冗余通道；

- BC 性能：
  - 支持 A/B 区域；
  - 具有自动重发功能；
  - 可编程的消息间隔时间；
  - 帧自动重复发送；
  - 可编程的超时响应时间；

- RT 性能：
  - 可编程的 RT 地址，子地址；
  - 支持单缓冲存储器管理方式；
  - 支持循环缓冲存储器管理方式；
  - 支持双缓冲存储器管理方式；
  - 可编程的非法命令表；
  - 可编程的方式代码中断表；
  - 可编程的子地址忙表；
- BM 性能：
  - 可选择的消息监测；
  - 支持命令堆栈半满、全满溢出；
  - 支持数据堆栈半满、全满溢出；

1.3 结构框图

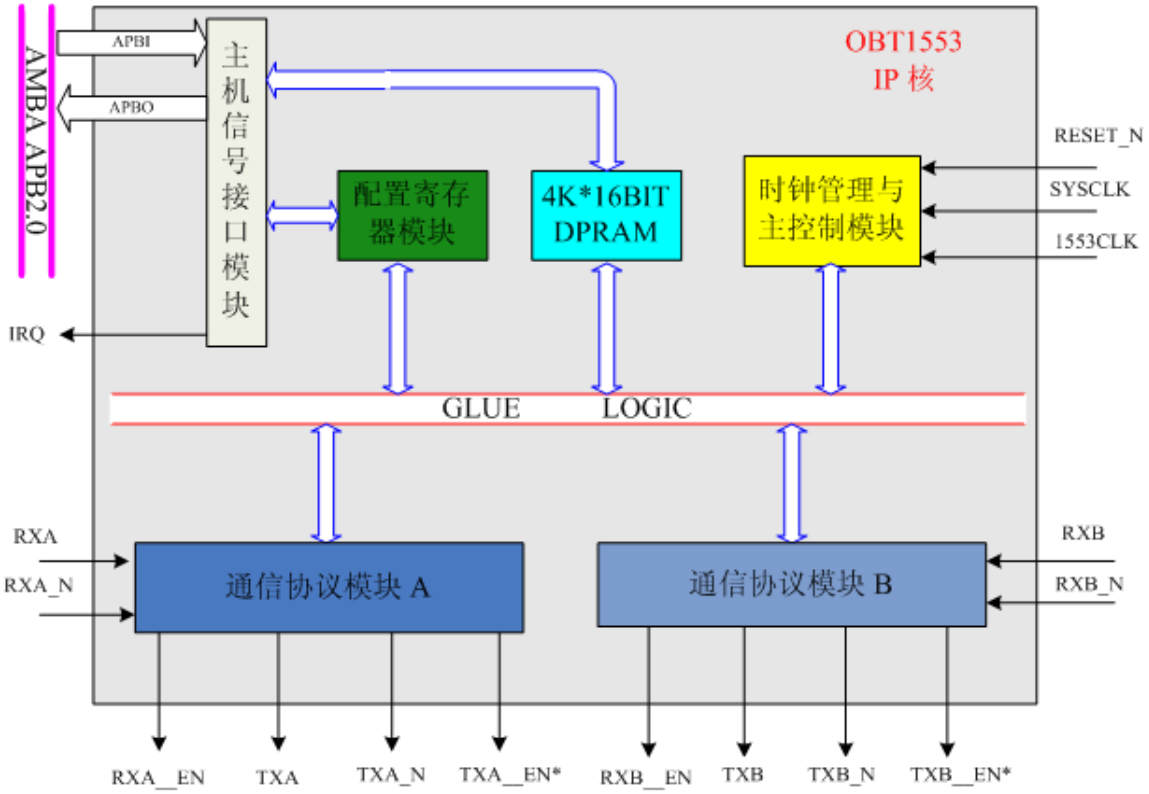

图 1-1 OBT1553B IP 核结构框图

OBT1553 IP 核结构框图如图 1-1 所示。各模块与信号说明分别如下表 1-1,表 1-2 所示：

表 1-1 OBT1553 IP 核各模块说明

序号	模块名称	模块描述
1	通信协议模块 A/B	通信协议模块部分用差分曼彻斯特编码实现时分命令响应式串行通讯，主要包括 A、B 通道编码器和解码器
2	主机信号接口模块	主机信号接口模块实现 1553 IP 核与 CPU 的接口，用来实现 CPU 对 OBT1553B IP 核的控制
3	配置寄存器模块	配置寄存器主要实现对 OBT1553 IP 核功能的配置，能够间接反映 1553 IP 核的功能

序号	模块名称	模块描述
4	存储管理模块（4K*16BIT DPRAM）	存储管理为 CPU 和 1553 IP 核之间交互数据的管理方式，主要为 4K*16BIT 的双口 RAM
5	时钟管理与主控制模块	时钟管理与主控制模块实现对输入时钟的管理和对 OBT1553 IP 核的基本通讯功能的实现与控制，是该 IP 核的中心控制单元能配置成 BC 或 RT 控制器

表 1-2 OBT1553B IP 核外部接口信号说明

序号	信号名称	信号方向	默认状态	信号描述
1	AMBA APB2.0	I/O	均为 0	主机接口接 AMBA.APB2.0 信号，主要是地址线，数据线，读写控制线
2	RXA	I	0	通道 A 接收信号
3	RXA_N	I	0	通道 A 接收信号的反
4	RXA_EN	0	0	通道 A 接收使能信号,主要为外接 HI-1567PSI 芯片时的使能
5	TXA	0	0	通道 A 发送信号
6	TXA_N	0	0	通道 A 发送信号的反
7	TXA_EN*	0	1	通道 A 发送使能信号，主要为外接 HI-1567PSI 芯片时的使能
8	RXB	I	0	通道 B 接收信号
9	RXB_N	I	0	通道 B 接收信号的反
10	RXB_EN	0	0	通道 B 接收使能信号,主要为外接 HI-1567PSI 芯片时的使能
11	TXB	0	0	通道 B 发送信号

序号	信号名称	信号方向	默认状态	信号描述
12	TXB_N	O	0	通道 B 发送信号的反
13	TXB_EN*	O	1	通道 B 发送使能信号，主要为外接 HI-1567PSI 芯片时的使能
14	RESET_N	I	1	外部复位信号低电平有效
15	SYSCLK	I	外部确定	主机输入时钟
16	1553CLK	I	16MHZ	1553 收发模块专用时钟
17	IRQ	O	0	中断信号输出，高电平有效

2. 主要功能模块概述

2.1 总线控制器（BC）

当主控制器配置为 BC 总线控制器时，则实现 BC 总线控制器功能。

BC 总线控制器控制通讯数据流的传输，是数据发送和接收的发起者和总线网络的管理者。控制计算机将数据写入 BC 总线控制器的内部存储器，并通过 开始/启动（SSR）寄存器来启动 BC 总线控制器进行数据传输。对于每个消息，BC 总线控制器通过 BC 控制字初始化 BC 总线控制器的状态从而发起数据传输（发送或者接收），并通过 BC 命令字通知 RT 响应数据传输（接收或者发送）。BC 总线控制器还可以通过模式命令对 RT 进行控制，包括读取同步和状态字等内容。

BC 总线控制器一方面通过 BC 模块状态字判断接受到的数据是否正确（包括奇偶校验）、响应是否超时等，另一方面通过读回的 RT 状态字，判断 RT 接收的数据是否正确、响应是否超时等。BC 模块状态字和读回的 RT 状态字均正常，说明数据传输正常。

如果传输过程中出现错误（BC 状态字和 RT 状态字异常），BC 总线控制器通过中断通知控制计算机进行处理，如消息重发。如果 BC 总线控制器出现灾难性故障，控制计算机（指 BC 的控制计算机）对 BC 复位。

2.2 远程终端（RT）

当主控制器配置为 RT 远程终端时，则实现 RT 远程终端功能。

RT 能根据协议在规定的时间内响应 BC 总线控制器发出的命令，进行数据接收或发送。RT 对输入信号进行检测，当检测到跟该 RT 地址一致的命令字后，响应数据传输。对于发送命令，RT 在发送数据之前将 RT 状态字通过 1553 总线发送给 BC；对于接收命令，RT 在接收完数据后将 RT 状态字通过 1553 总线发送给 BC。BC 通过 RT 状态字判断本次数据传输（发送/接收）是否有效。

当接受到模式命令后，RT 需要对接受到的模式命令进行响应。

2.3 总线监视器（BM）

BM 能够实时侦听总线上的数据流，可以将所有的数据流记录下来，也可以有选择地进行数据监听。并设有命令存储区半满、全满标志和数据存储区半满、全满标志。

3. 寄存器描述

3.1 寄存器地址分配表

表 3-1　寄存器地址分配

地址（HEX）	读/写	有效位宽	默认值(HEX)	寄存器描述
0x00	RD/WR	16	0000	中断屏蔽寄存器(IMR)
0x01	RD/WR	16	0000	配置寄存器 1(CFG1)
0x02	RD/WR	16	0000	配置寄存器 2(CFG2)
0x03	WR	16	0000	启动/复位寄存器(SRR)
0x03	RD	16	0000	BC/RT/BM 命令堆栈指针寄存器(STACK_ADDR)
0x04	RD/WR	16	0000	BM 初始命令堆栈指针寄存器
0x05	RD	16	0000	时间标签寄存器 0(TTR0)
0x06	RD	16	0000	中断状态寄存器(INT_STA)
0x07	RD/WR	16	0000	配置寄存器 3 (CFG3)
0x08	RD/WR	16	0000	配置寄存器 4(CFG4)
0x09	RD/WR	16	0000	配置寄存器 5(CFG5)
0x0A	RD	16	0000	BM 数据堆栈指针寄存器（BM_STACK_ADDR）
0x0B	RD	16	0000	1Mbps/10Mbps 配置寄存器
0x0C	RD	16	0000	保留
0x0D	RD	16	0000	RT 上一命令字寄存器(LAST_CMD)
0x0D	WR	16	0000	BC 帧时间寄存器
0x0E	RD	16	0000	RT 状态字寄存器（RT_STA）

0x0F	RD	16	0000	RT BIT 字寄存器(RT_BIT_REG)
0x10	RD	16	0000	时间标签寄存器 1(TTR1)
0x11	RD	16	0000	时间标签寄存器 2(TTR2)
0x12	RD/WR	16	0000	空消息寄存器

3.2 中断屏蔽寄存器（IMR）

表 3-2 中断屏蔽寄存器（IMR）

位（BIT）	描述 (DESCRIPTION)
15	保留
14	RAM 奇偶校验错误（RAM PARITY ERROR）
13	BC/RT 传输器超时（BC/RT TRANSMITTER TIMEOUT）
12	BC/RT 命令堆栈溢出/ BM 命令堆栈半满溢出
11	BM 命令堆栈溢出（BM COMMAND STACK ROLLOVER）
10	BM 数据堆栈溢出（BM DATA STACK ROLLOVER）
9	保留
8	BC 重发/BM 数据半满溢出
7	RT 地址奇偶校验错误（RT ADDRESS PARITY ERROR）
6	时间标签寄存器溢出（TIME TAG ROLLOVER）
5	RT 循环缓存溢出（RT CIRCULAR BUFFER ROLLOVER）
4	BC 消息/RT 子地址控制字消息结束（BC MSG/RT SUBADDRESS CONTROL WORD EOM）
3	BC 帧结束（BC END OF FRAME）
2	格式错误(FORMAT ERROR)
1	BC 状态置位/RT 方式代码（BC STATUS SET/RT MODE CODE）
0	消息结束（END OF MESSAGE）

3.3 BC配置寄存器 1(BC-CFG1)

表 3-3 BC 配置寄存器 1(BC-CFG1)

位（BIT）	描述 (DESCRIPTION)
15(MSB)	BC/RT/BM 模式设置（RT/BM*-BC*）
14	BC/RT/BM 模式设置（BM/RT*- BC*）
13-9	A*/B 区域设置（CURRENT AREA A*/B）
12-9	保留
8	帧自动重复发送使能（FRAME AUTO-REPEAT）
7-6	保留
5	消息间隔时间使能（MESSAGE GAP TIMER ENABLED）
4	消息重发使能（RETRY ENABLED）
3	消息重发一次或二次选择（DOUBLE/SINGLE* RETRY）
2	BC 使能（BC ENABLED），*该位只读*
1	BC 帧信息忙指示（BC FRAME IN PROGRESS)，*该位只读*
0 (LSB)	BC 消息忙指示（BC MESSAGE IN PROGRESS），*该位只读*

3.4 RT配置寄存器 1 (RT-CFG1)

表 3-4 RT 配置寄存器 1(RT- CFG1)

位（BIT）	描述 (DESCRIPTION)
15(MSB)	BC/RT/BM 模式设置（RT/BM*-BC*）
14	BC/RT/BM 模式设置（BM/RT*- BC*）
13-12	保留
11	动态总线控制接收*（DYNAMIC BUS CONTROL ACCEPTANCE*）
10	忙*（BUSY*）
9	服务请求*（SERVICE REQUEST*）
8	子系统标志*（SUBSYSTEM FLAG*）

位（BIT）	描述 (DESCRIPTION)
7	RT 标志*（RTFLAG*）
6-1	保留
0	BC 消息忙指示（BC MESSAGE IN PROGRESS），*该位只读*

3.5 配置寄存器 2 (CFG2)

表 3-5 配置寄存器 2(CFG2)

位（BIT）	描述 （DESCRIPTION）
15-14	保留
13	忙查找表使能(BUSY LOOK UP TABLE ENABLE)
12-10	保留
9-7	时间标签最小精度设置（TIME TAG RESOLUTION2，1，0）
6	同步清除时标寄存器使能（CLEAR TIME TAG ON SYNCHRONIZE）
5	同步重载时标寄存器使能（LOAD TIME TAG ON SYNCHRONIZE）
4	中断状态自动清除（INTERRUPT STATUS AUTO CLEAR）
3	电平/脉冲中断(LEVEL/PULSE *INTERRUPT REQUEST)
2	清除服务请求(CLEAR SERVICE REQUEST)
1-0	保留（其中 BIT1 可进行读写，但没有实际意义）

3.6 启动/复位寄存器（SRR)

表 3-6 启动/复位寄存器（SRR）

位（BIT）	描述 (DESCRIPTION)
15-7	保留
6	BC 停止消息发送（BC STOP-ON-MESSAGE）
5	BC 停止帧发送（BC STOP-ON-FRAME）
4	保留
3	时间标签寄存器清零（TIME TAG RESET）

位（BIT）	描述 (DESCRIPTION)
2	中断状态寄存器清零（INTERRUPT RESET）
1	BC/BM 启动（BC/BM START）
0	系统软复位（RESET）

3.7 BC/RT/BM命令堆栈指针寄存器（STACK_ADDR)

表 3-7 BC/RT 命令堆栈指针寄存器(STACK_ADDR)

位（BIT）	描述 （DESCRIPTION）
15-0	BC/RT/BM 命令堆栈指针

3.8 BM初始命令堆栈指针寄存器（INIT_STACK_ADDR)

表 3-8 BM 初始命令堆栈指针寄存器（INIT_STACK_ADDR)

位（BIT）	描述 (DESCRIPTION)
15-0	BM 命令堆栈指针初始位置

3.9 时间标签寄存器 0（TTR0）

表 3-9 时间标签寄存器 0（TTR）

位（BIT）	描述 （DESCRIPTION）
15-0	时间计时标签位 BIT15-BIT0

3.10 中断状态寄存器（INT_STA）

表 3-10 中断状态寄存器（INT_STA）

位（BIT）	描述 (DESCRIPTION)
15	中断请求（MASTER INTERRUPT）
14	保留
13	BC/RT 传输器超时（BC/RT TRANSMITTER TIMEOUT）
12	BC/RT 命令堆栈溢出/BM 命令堆栈半满溢出
11	BM 命令堆栈溢出（BM COMMAND STACK ROLLOVER）

位（BIT）	描述 (DESCRIPTION)
10	BM 数据堆栈溢出（BM DATA STACK ROLLOVER）
9	保留
8	BC 重发（BC RETRY）/BM 数据半满溢出
7	RT 地址奇偶校验错误（RT ADDRESS PARITY ERROR）
6	时间标签寄存器溢出（TIME TAG ROLLOVER）
5	RT 循环缓存溢出（RT CIRCULAR BUFFER ROLLOVER）
4	BC 消息/RT 子地址控制字消息结束（BC MSG/RT SUBADDRESS CONTROL WORD EOM）
3	BC 帧结束（BC END OF FRAME）
1	BC 状态置位/RT 方式代码（BC STATUS SET/RT MODE CODE）
0	消息结束（END OF MESSAGE）

3.11 配置寄存器 3（CFG3）

表 3-11 配置寄存器 3(CFG3)

位（BIT）	描述 (DESCRIPTION)
15-8	保留
7	非法命令查找表使能（ILLEGALIZATION DISABLED）
3	接收忙屏蔽(BUSY RX TRANSFER ENABLE)
2-1	保留
0	增强方式代码功能(ENHANCED MODE CODE HANDLING)

3.12 配置寄存器 4(CFG4)

表 3-12 配置寄存器 4(CFG4)

位（BIT）	描述 (DESCRIPTION)
15-9	保留
8	第一次重发通道选择（FIRST RETRY ALT/SAME* BUS）

位（BIT）	描述 (DESCRIPTION)
7	第二次重发通道选择（SECOND RETRY ALT/SAME* BUS）
6-4	保留
3	RT 地址配置使能(LATCH RT ADDRESS WITH CFG REG #5)
2-0	保留

3.13 配置寄存器 5 (CFG5)

表 3-13 配置寄存器 5(CFG5)

位（BIT）	描述 (DESCRIPTION)
15-11	保留
10-9	超时响应时间设置(RESPONSE TIMEOUT SELECT1,0)
8-6	保留（其中 BIT6 可进行读写，但没有实际意义）
5-1	RT 地址位 4-0(RT ADDRESS4-ADDRESS0)
0	RT 地址奇偶位(RT ADDRESS PARITY)

3.14 BM数据堆栈指针寄存器（BM_STACK_ADDR)

表 3-14 BM 数据堆栈指针寄存器（BM_STACK_ADDR)

位（BIT）	描述 (DESCRIPTION)
15-0	BM 数据堆栈指针

3.15 1Mbps/10Mbps配置寄存器（1M_10M_SEL)

表 3-15 1Mbps/10Mbps 配置寄存器（1M_10M_SEL)

位（BIT）	描述 (DESCRIPTION)
15-1	保留
0	1Mbps/10Mbps 设置位

3.16 BC帧时间/RT上一命令字寄存器(LAST_CMD)

表 3-16 BC 帧时间/RT 上一命令字寄存器（LAST_CMD)

位（BIT）	描述 (DESCRIPTION)
15-0	BC 帧时间/RT 上一命令字寄存器

3.17 RT状态字寄存器(RT_STA)

表 3-17 RT 状态字寄存器(RT_STA)

位（BIT）	描述 (DESCRIPTION)
15-11	均为 0
10	消息错误(MESSAGE ERROR)
9	测试手段(INSTRUMENTATION)
8	服务请求(SERVICE REQUEST)
7-5	保留
4	广播指令接收(BROADCAST COMMAND RECEIVED)
3	忙(BUSY)
2	子系统标志(SYBSYSTEM FLAG)
1	动态总线控制接收(DYNAMIC BUS CONTROL ACCEPT)
0	终端标志(TERMINAL FLAG)

3.18 RT BIT字寄存器(RT_BIT_REG)

表 3-18 RT BIT 字寄存器(RT_BIT_REG)

位（BIT）	描述 (DESCRIPTION)
15	传输器超时（TRANSMITTER TIMEOUT）
14-12	保留
11	通道 B 发送器关闭（TRANSMITTER SHUTDOWN B）
10	通道 A 发送器关闭（TRANSMITTER SHUTDOWN A）
9	终端标志禁止(TERMINAL FLAG INHIBITED)
8	总线传输通道 B/A*(CHANNEL B/A*)

位（BIT）	描述 (DESCRIPTION)
7	保留
6	字计数错（WORD COUNT ERROR）
5	错误数据同步头（INCORRECT SYNC RECEIVED）
4	奇偶/位计数错(PARITY/BIT COUNT ERROR)
3	RT-RT 同步头/地址错(RT-RT SYNC/ADDRESS ERROR)
2	RT-RT 响应超时（RT-RT NO RESPONSE ERROR)
1	RT-RT 第二个命令字错(RT-RT 2ND COMMAND WORD ERROR)
0	命令字内容错(COMMAND WORD CONTENTS ERROR)

3.19 时间标签寄存器 1（TTR1）

表 3-19 时间标签寄存器 1（TTR1）

位（BIT）	描述 （DESCRIPTION）
15-0	时间计时标签位

3.20 时间标签寄存器 2（TTR2）

表 3-20 时间标签寄存器 2（TTR2）

位（BIT）	描述 （DESCRIPTION）
15-0	时间计时标签位

3.21 空消息寄存器（IDLE_REG）

表 3-21 空消息寄存器（IDLE_REG）

位（BIT）	描述 (DESCRIPTION)
15-14	保留
13-8	表示插入空消息的起始位置
7-6	保留
5-0	表示插入的空消息条数

3.22 BC控制字（BC_CTRL）

表 3-22 BC 控制字（BC_CTRL）

位（BIT）	描述 (DESCRIPTION)
15	保留
14	消息格式错误屏蔽（M.E.MASK）
13	服务请求位屏蔽(SERVICE REQUEST BIT MASK)
12	忙位屏蔽(SUBSYS BUSY BIT MASK)
11	子系统标志位屏蔽(SUBSYS FLAG BIT MASK)
10	终端标志位屏蔽(TERMINAL FLAG BIT MASK)
9	保留
8	重试使能(RETRY ENABLED)
7	总线通道选择 A/B*(bus channel a/b*)
6	保留
5	保留
4	EOM 中断使能(EOM INTERRUPT ENABLE)
3	保留
2	模式命令(MODE CODE FORMAT)
1	广播命令(BROADCAST FORMAT)
0	RT2RT(RT-TO-RT FORMAT)

3.23 BC命令字（BC_CMD）

表 3-23 BC 命令字(BC_CMD)

位（BIT）	描述 (DESCRIPTION)
15-11	远程终端地址（REMOTE TERMINAL ADDRESS）
10	发送/接收* (T/R*)
9-5	子地址/方式(SUBADDRESS/MODE)
4-0	数据字计数/方式代码(DATA WORD COUNT/MODE CODE)

3.24 BC块状态字(BC_BLK)

表 3-24 BC 块状态字(BC_BLK)

位（BIT）	描述 (DESCRIPTION)
15	消息传输结束标志位（EOM）
14	保留
13	传输通道指示(CHANNEL B/A*)
12	出错标志(ERROR FLAG)
11	状态设置（STATUS SET）
10	格式错误(FORMAT ERROR)
9	响应超时(NO RESPONSE TIMEOUT)
8	保留
7	屏蔽状态设置（MASKED STATUS SET）
6-5	重发消息次数(RETRY COUNT 1,0)
4	数据传输正常（GOOD DATA BLOCK TRANSFER）
3	错误的状态地址（WRONG STATUS ADDRESS）
2	字计数错误（WORD COUNT ERROR）
1	同步头出错(INCORRECT SYNC TYPE)
0	无效字（INVALID WORD）

3.25 RT子地址控制字(RT_SUB_CTRL)

表 3-25 RT 子地址控制字(RT_SUB_CTRL)

位（BIT）	描述 (DESCRIPTION)
15	RX: 接收双缓存使能（DOUBLE BUFFER ENABLE）
14	TX: 发送消息结束中断使能（EOM INT）
13	TX: 发送循环缓存溢出使能(CIRC BUF INT)
12-10	TX: 发送存储器管理设置（MEMORY MANAGEMENT2,1,0）
9	RX: 接收消息结束中断使能（EOM INT）
8	RX: 接收循环缓存溢出使能(CIRC BUF INT)

位（BIT）	描述 (DESCRIPTION)
7-5	RX: 接收存储器管理设置（MEMORY MANAGEMENT2,1,0）
4	BCST: 广播消息结束中断使能（EOM INT）
3	BCST: 广播循环缓存溢出使能(CIRC BUF INT)
2-0	BCST: 广播存储器管理设置（MEMORY MANAGEMENT2,1,0）

3.26 RT/BM块状态字(RT/BM_BLK)

表 3-26 RT/BM 块状态字(RT/BM_BLK)

位（BIT）	描述 (DESCRIPTION)
15	消息传输结束标志（EOM）
14	消息传输开始标志（SOM）
13	传输通道指示（CHANNEL B/A*）
12	出错标志(ERROR FLAG)
11	RT-RT 通讯方式标志(RT-RT FORMAT)
10	格式错误标志(FORMAT ERROR)
9	响应超时标志(NO RESPONSE TIMEOUT)
8	保留
7	循环缓存溢出（DATA STACK ROLLOVER）
6-5	非法命令字(ILLEGAL COMMAND WORD)
4	字计数个数错(WORD COUNT ERROR)
3	数据同步头出错(INCORRECT DATA SYNC)
2	无效字（INVALID WORD）
1	RT-RT 同步头/地址错（RT-RT SYNCH/ADDRESS ERROR）
0	RT-RT 第二个命令字错（RT-RT 2ND COMMAND ERROR）

4. 功能模块描述

4.1 BC总线控制器工作方式

4.1.1 BC存储器地址分配

表 4-1 BC 存储器地址分配(4K 双口 RAM)

地址（HEX）	描述
0000-00FF	堆栈 A（STACK A）
0100	堆栈指针 A (STACK POINTER A)
0101	消息个数设置 A (MESSAGE COUNT A)
0102-0103	保留
0104	堆栈指针 B (STACK POINTER B)
0105	消息个数设置 B (MESSAGE COUNT B)
0106-0107	保留
0108-012D	消息块 0(MESSAGE BLOCK0)
012E-0153	消息块 1(MESSAGE BLOCK1)
…	…
0ED6-0EFB	消息块 93(MESSAGE BLOCK 93)
…	…
0F00-0FFF	堆栈 B（STACK B）

4.1.2 BC存储器管理

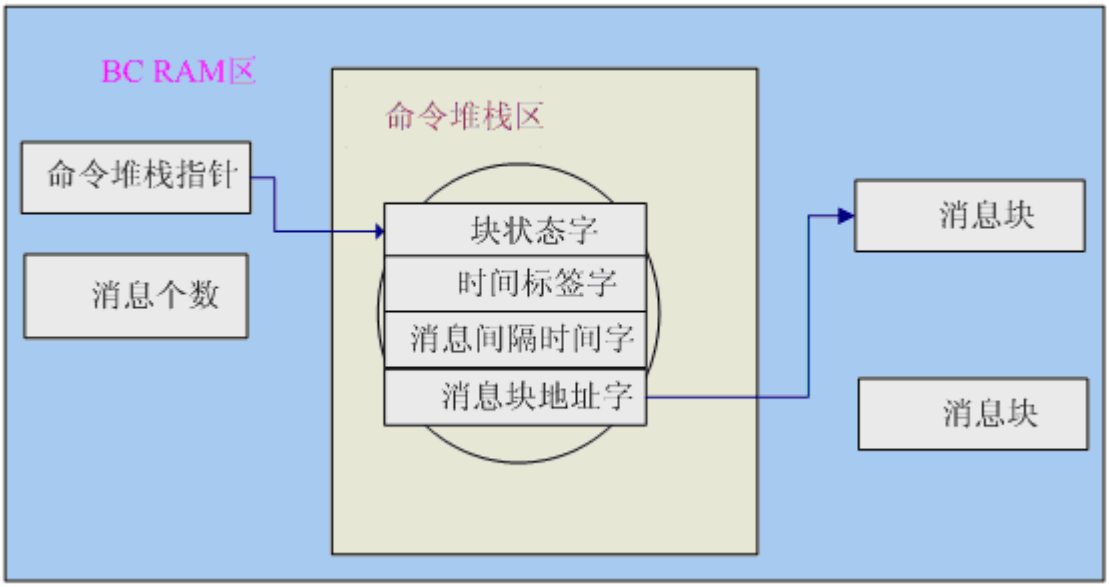

图 4-1 BC 存储器管理

4.1.3 BC消息格式

表 4-2 BC 消息格式

BC 到 RT 的传输	RT 到 BC 的传输	RT 到 RT 的传输	不带字的方式命令
控制字	控制字	控制字	控制字
接收命令字	发送命令字	接收命令字	方式命令字
数据字 1	发送命令字的回应字	发送命令字	方式回应字
数据字 2	状态字	发送命令字的回应字	状态字
…	数据字 1	发送终端的状态字	
	数据字 2	数据字 1	
最后一个数据字	…	…	
最后数据字的回应字		最后一个数据字	

状态字	最后一个数据字	接收终端的状态字	

表 4-3 BC 消息格式（接上表）

带字的发送方式命令	带字的接收方式命令	广播命令	不带字的广播方式命令	带字的广播方式命令
控制字	控制字	控制字	控制字	控制字
发送方式命令字	接收方式命令字	广播命令字	广播方式命令字	广播方式命令字
方式命令回应字	数据字	数据字 1	广播方式命令字的回应字	数据字
接收状态字	接收命令字的回应字	数据字 2 …		广播方式命令字的回应字
数据字	接收状态字	最后的数据字		

4.2 RT远程终端工作方式

4.2.1 RT存储器地址分配

表 4-4 RT 存储器地址分配(4K 双口 RAM)

地址（HEX）	描述
0000-00FF	堆栈 (STACK)
0100	堆栈指针(STACK POINTER)
0101-0107	保留
0108-010F	方式代码选择中断表（MODE CODE SELECTIVE INTERRUPT TABLE）
0110-013F	方式代码数据(MODE CODE DATA)
0140-01BF	查找表(LOOKUP TABLE)
01C0-023F	保留
0240-0247	忙位查找表（BUSY BIT LOOKUP TABLE）
0248-025F	（没有使用）

地址（HEX）	描述
0260-027F	数据块 0 (DATA BLOCK 0)
0280-02FF	数据块 1-4　(DATA BLOCK 1-4)
0300-03FF	非法命令表 (CINNABD UKKEGAKUZUBG TABKE)
0400-041F	数据块 5 (DATA BLOCK 5)
0420-043F	数据块 6 (DATA BLOCK 6)
…	…
0FE0-0FFF	数据块 100 (DATA BLOCK 100)

4.2.2 RT存储器查找表

表 4-5　RT 存储器查找表(LOOK_UP TABLE)

地址（HEX）	对应子地址	描述
0140 … 015F	Rx_SA0 … Rx _SA31	接收查找表
0160 … 017F	Tx_SA0 … Tx _SA31	发送查找表
0180 … 19F	Bcst_SA0 … Bcst _SA31	广播查找表
01A0 … 01BF	SACW_SA0 … SACW_SA31	子地址控制字查找表

4.2.3 IP核实现的方式代码

表 4-6 IP 核实现的方式代码

发/收*	方式代码	功能	是否带数据字	是否允许广播指令
1	00000	动态总线控制	否	否
1	00001	同步	否	是
1	00010	发送上一状态字	否	否
1	00011	启动自测试	否	是
1	00100	发送器关闭	否	是
1	00101	取消发送器关闭	否	是
1	00110	禁止终端标志位	否	是
1	00111	取消禁止终端标志位	否	是
1	01000	复位远程终端	否	是
1	01001	备用	否	待定
…	…	…	…	…
1	01111	备用	否	待定
1	10000	发送矢量字	是	否
0	10001	同步	是	是
1	10010	发送上一指令字	是	否
1	10011	发送自检测字	是	否

4.2.4 RT单缓冲存储器管理

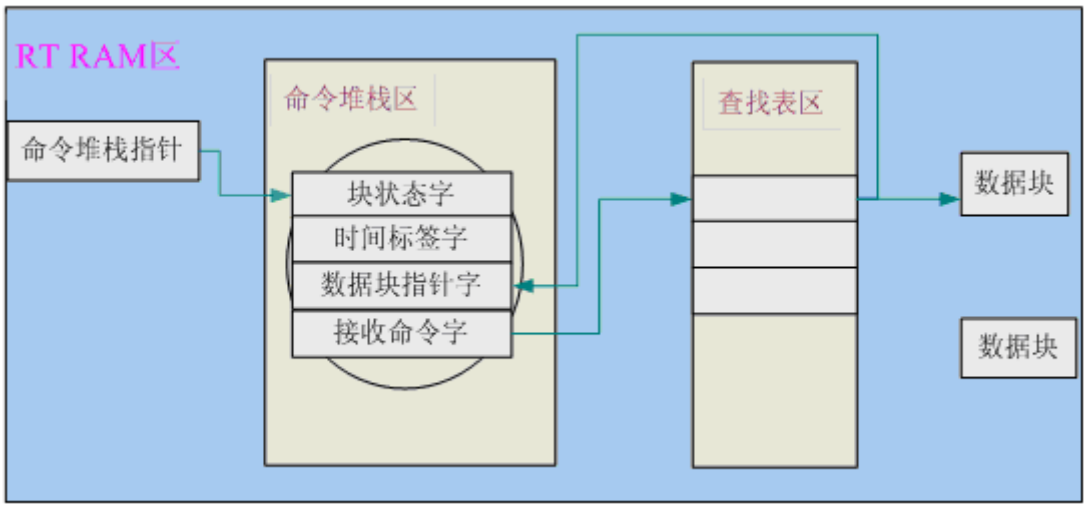

图 4-2 RT 单缓冲存储器管理

4.2.5 RT循环缓冲存储器管理

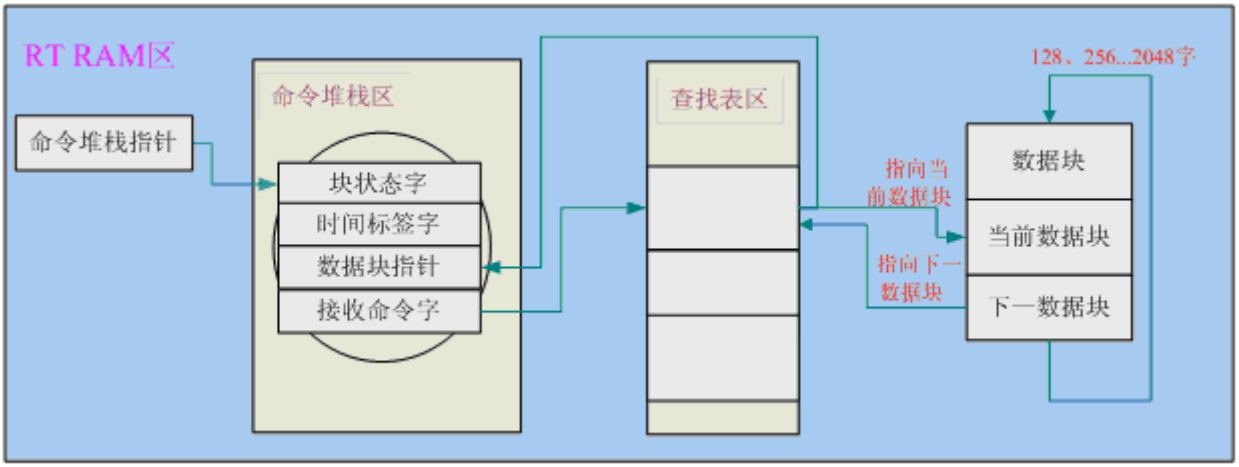

图 4-3 RT 循环缓冲存储器管理

4.2.6 RT双缓冲存储器管理

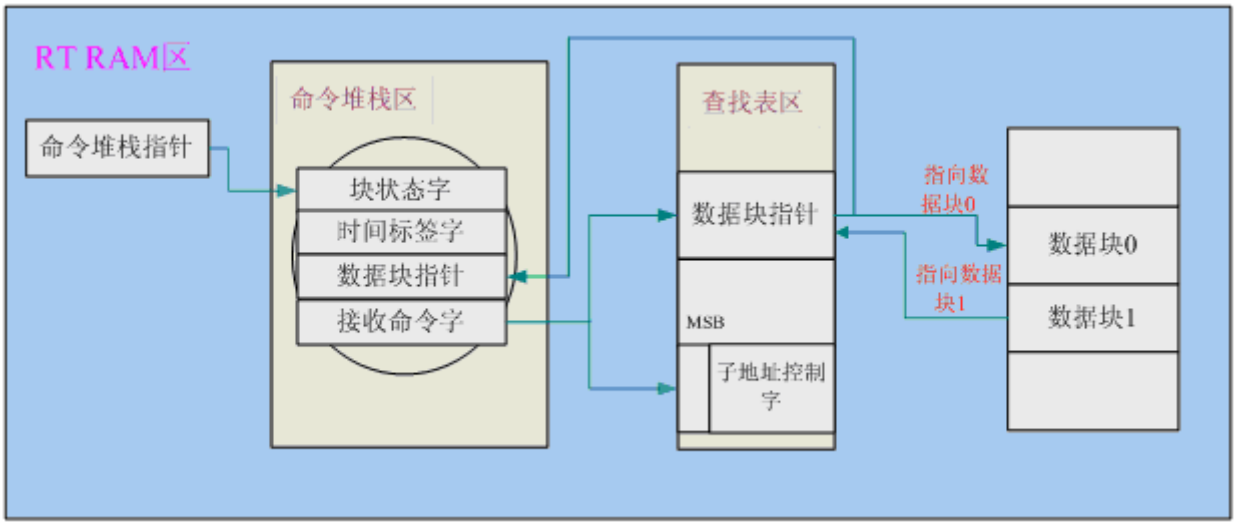

图 4-4 RT 双缓冲存储器管理

4.3 BM总线监视器工作方式

4.3.1 BM存储器地址分配

表 4-7 BM 存储器地址分配

地址（HEX）	描述
0000-027F	命令堆栈区域 (STACK AREA)
0280-02FF	子地址选择设置区域(SUBADDRESS SELECT AREA)
0300-0FFF	数据块区域(DATA BLOCK AREA)

4.3.2 BM存储器管理

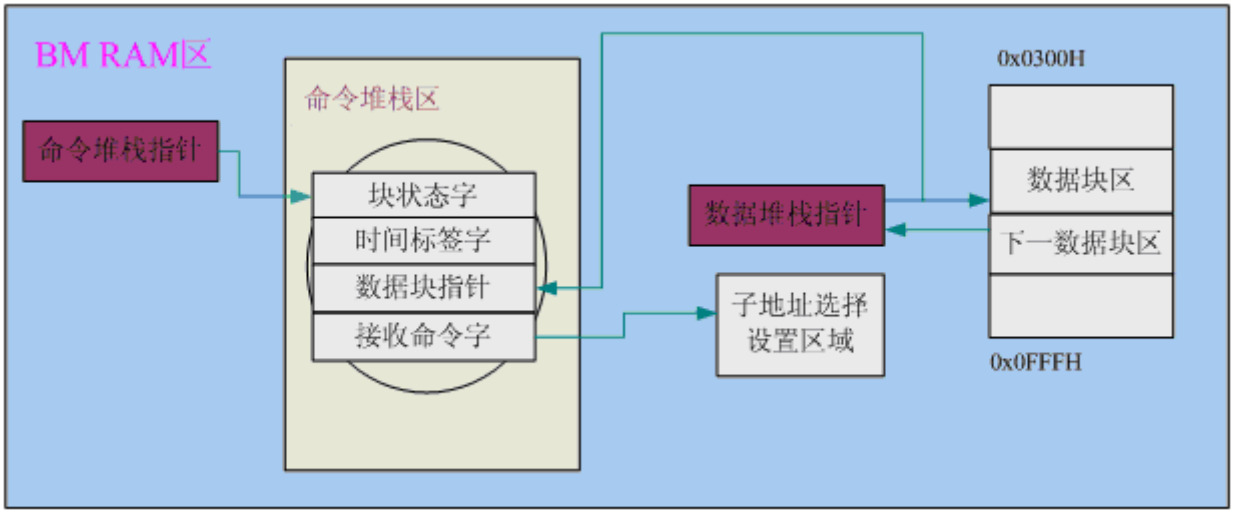

图 4-5 BM 存储器管理

4.3.3 BM子地址选择设置区地址分配

位（BIT）	描述 (DESCRIPTION)
15-11	逻辑 0
10	逻辑 0
9	逻辑 1
8	逻辑 0
7	逻辑 1
6	RT 地址 4（RT ADDRESS 4）
5	RT 地址 3（RT ADDRESS 3）
4	RT 地址 2（RT ADDRESS 2）
3	RT 地址 1（RT ADDRESS 1）
2	RT 地址 0（RT ADDRESS 0）
1	发送/接收*（T/R*）
0	子地址 4（SA4）

5. 时序图

发送波形如 5-1 所示，接收波形如 5-2 所示，数据频率编码关系如 5-3 所示：

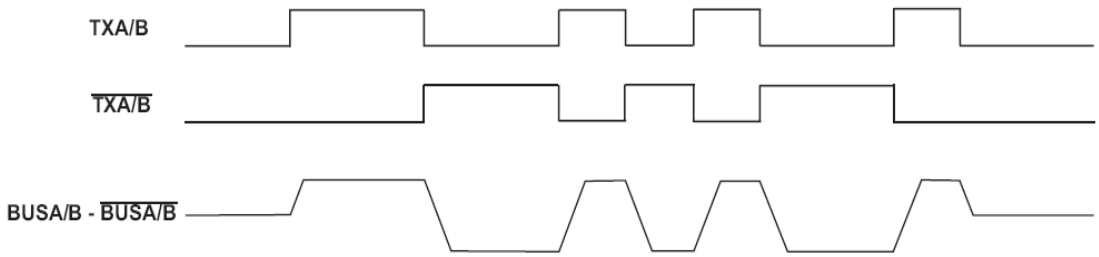

图 5-1 发送波形

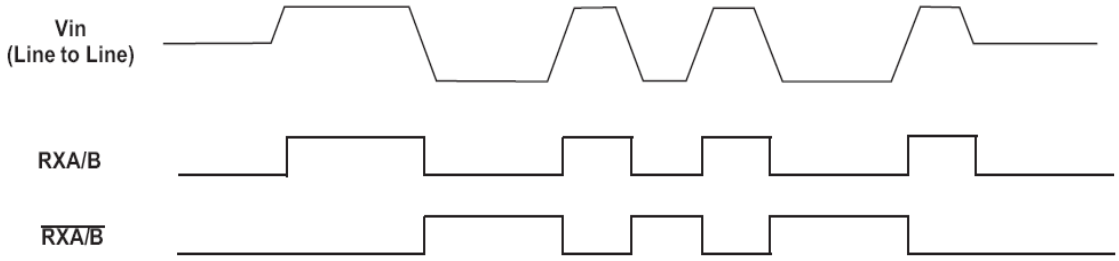

图 5-2 接收波形

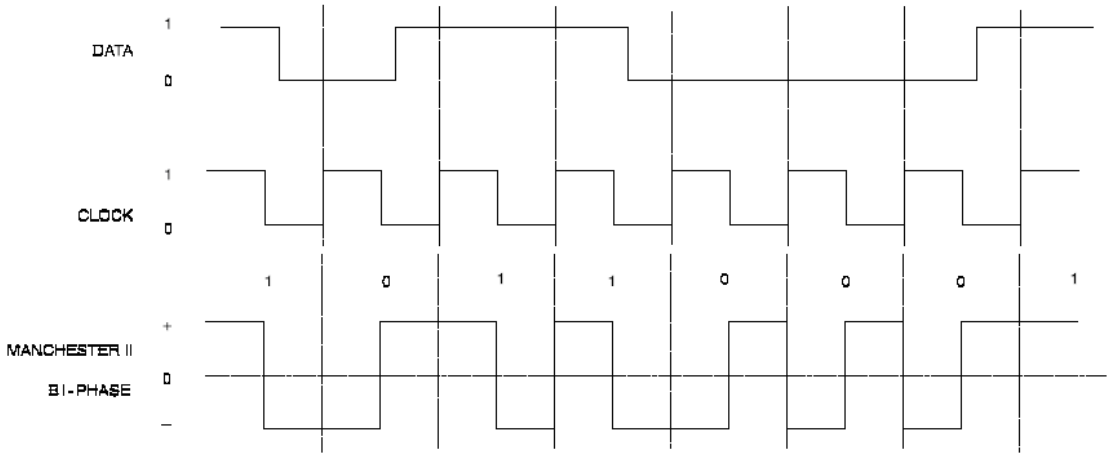

图 5-3 数据频率编码之间的关系

6. 应用案例

6.1 1Mbps外围接口

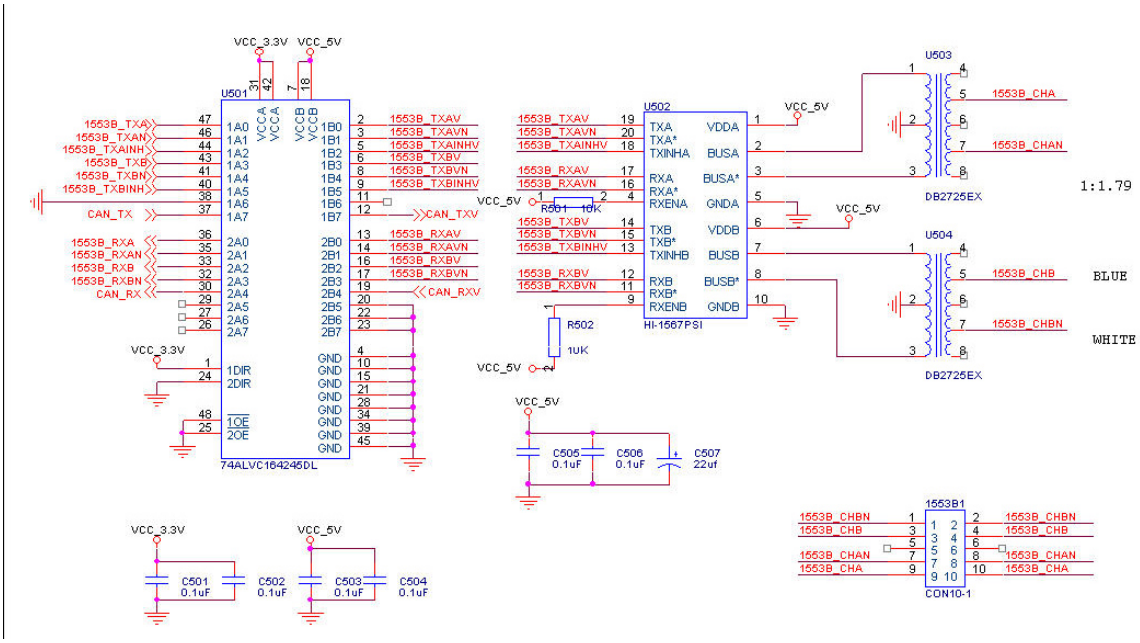

图 6-1 OBT1553-1M 外围接口原理图

OBT1553 外围接口原理图主要说明 1553 总线外部接线关系。因为 IP 集成后输出电平为 LVTTL 电平，而 HI-1567 芯片为 TTL 电平，中间需要通过一个电平转换芯片如 74ALVC164245DL 芯片进行电平转换，根据 1553 总线协议要求 1553 对外接口还要接变压器。

6.2 10Mbps外围接口

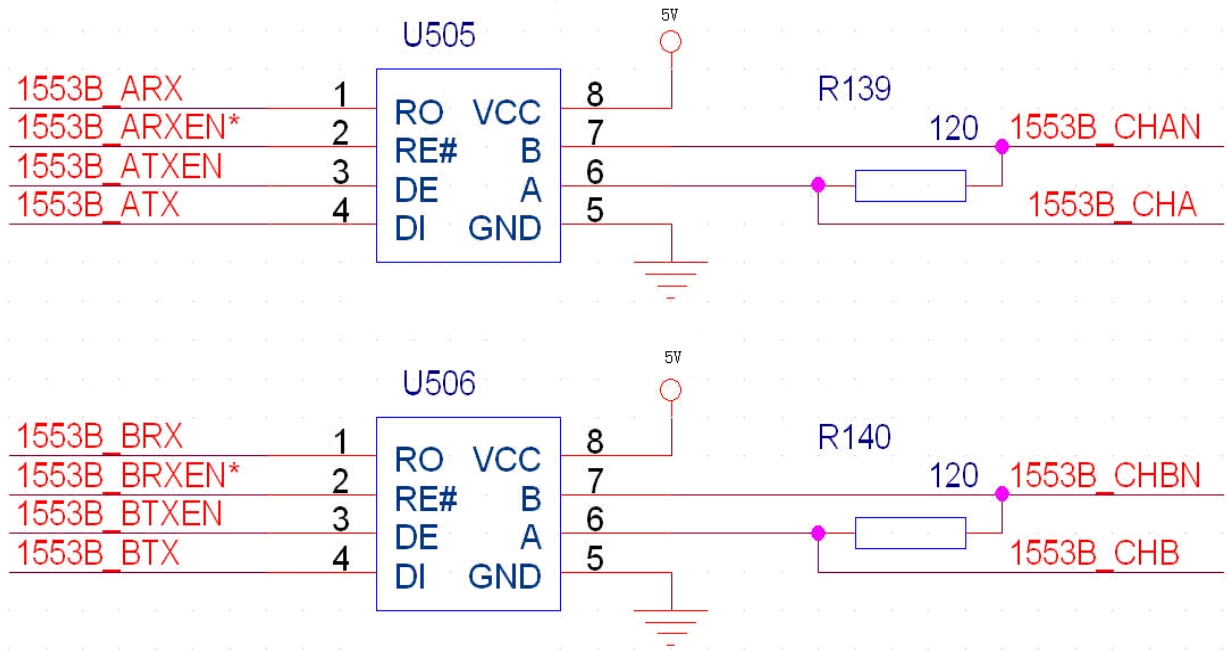

图 6-2　OBT1553-10M 接口原理图

OBT1553-10Mbps 外围接口原理图主要说明 1553-10M 总线外部接线关系。通过外接 RS485 芯片（图中的 U505，U506），进行差分信号传输。1553-10M 对外接口不接变压器。

6.3 BC总线控制器应用案例

对于 BC 编程，首先要初始化相应的寄存器以及堆栈指针、消息计数器；然后定义消息的控制字、命令字等；最后启动 BC。需要注意的是 BC 控制字不会在 1553 总线上传输。BC 的消息格式通过编程 BC 控制字的最低 3 位来控制。

BC 消息帧可以通过查询和中断来进行处理。如果采用查询模式，那么可以查询配置寄存器 1、中断状态寄存器、堆栈指针和消息计数器寄存器。另外，每一条 BC 消息结束后堆栈指针加 4。在嵌入式系统软件处理中，我们应尽量采用中断方式。

流程图如下图所示。程序把 OBT1553 设置成 BC 总线控制器类型，并且设置消息类型为 BC-RT，RT-BC；一帧两条消息，第一条消息为 BC 往地址为 0，子地址为 5 发送 32 个数据；第二条消息为地址 0，子地址 4 往 BC 发送 32 个数据。

BC 示例程序见附录一。

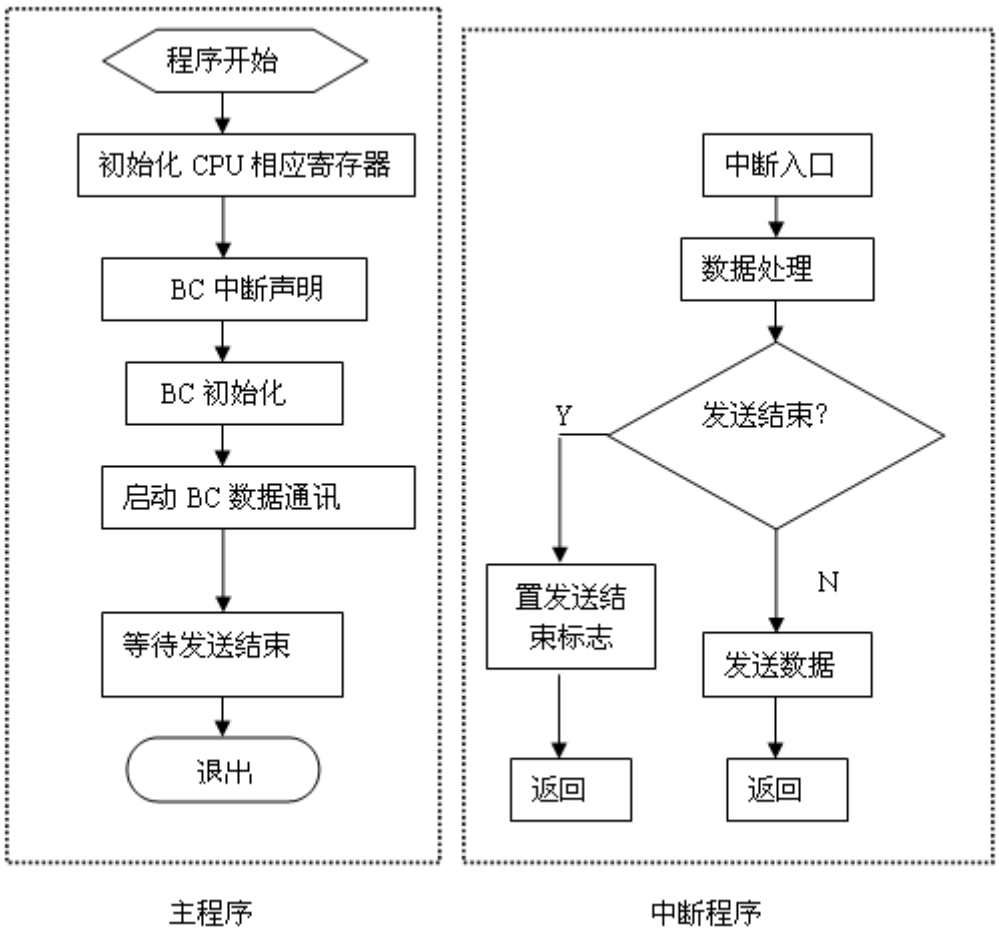

图 6-3 OBT1553 BC 程序流程图

6.4 RT远程终端应用案例

对于 RT 编程，首先初始化相应的寄存器；然后设置非法区、初始化相应子地址的查询表及子地址控制字；最后设置配置寄存器 1 使设备处于 RT 模式。此后该设备就处于在线，只要 BC 发送一条消息命令与该设备相关，那么该设备就会做出反映。处理 RT 消息时，这里也有四个字的块描述符，即块状态字、时间标志字、数据块起始地址指针和接收到的 16 位命令字。与 BC 模式一样，要读取接收到的消息，我们应该首先从堆栈指针中读取当前消息的堆栈指针，来分别读出块状态字、时间标志字、上一条消息的块地址和命令字。

流程图如下图所示。程序把 OBT1553 设置成 RT 远程终端类型，并且设置消息类型为 BC-RT；BC 往 RT 地址 0，子地址为 1, MC32 发送消息，消息的数据长度为 1 个，数据为互为反码的一组数----0x0000 和 0xffff。

RT 示例程序见附录二。

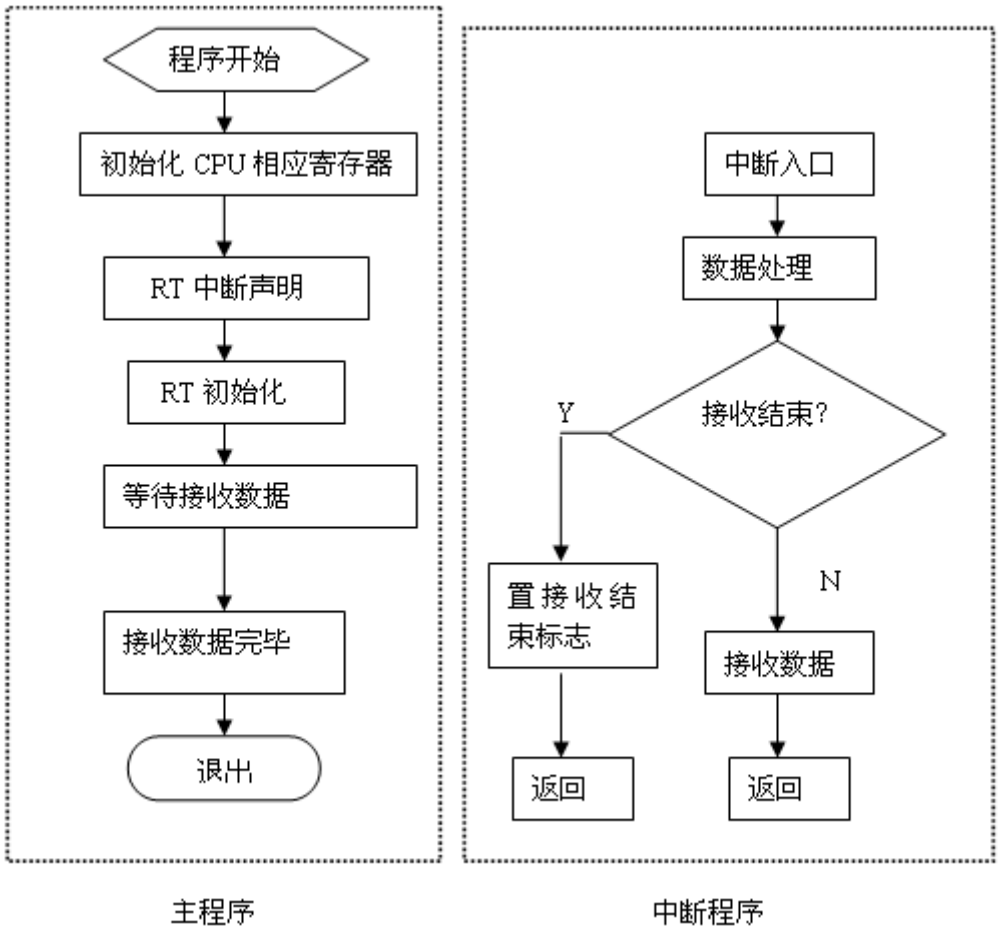

图 6-4 OBT1553 RT 程序流程图

6.5 BM总线监视器应用案例

对于 BM 编程，首先初始化相应的寄存器；然后设置子地址选择设置区；最后设置启动/复位寄存器启动 BM。此后该设备就处于在线。接收到的数据里也有四个字的块描述符，即块状态字、时间标志字、数据块起始地址指针和接收到的 16 位命令字。与 RT 模式一样，要读取接收到的消息，我们应该首先从堆栈指针中读取当前消息的堆栈指针，来分别读出块状态字、时间标志字、上一条消息的块地址和命令字; 从数据堆栈指针中读取接收到的数据。

流程图如下图所示。程序把 OBT1553B 设置成 BM 类型，接收消息类型 BC→RT0 SA1 MC32 ,判断 BM 接收的数据是否是 BC 发送的数据。

BM 示例程序见附录三。

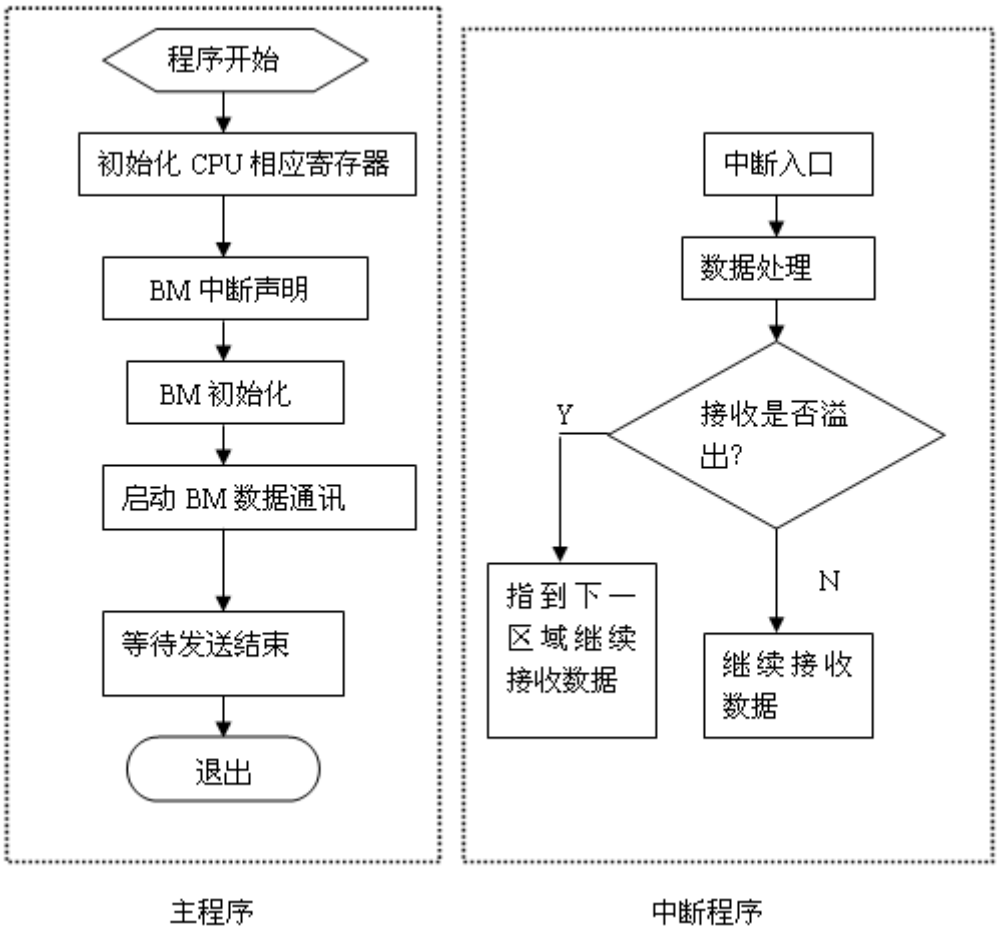

图 6-5 OBT1553 BM 程序流程图

7. 附录

7.1 附录一：BC示例程序

Bc.h

```
#include<stdio.h>
#define IRQMP_BASE                0x80000200
#define   INTMASK       *(volatile unsigned int *)(IRQMP_BASE+0x40)   //R/W   FIRST INT MASK REGISTER
#define   INTPEND       *(volatile unsigned int *)(IRQMP_BASE+0x4)   //R   FIRST INT PEND REGISTER
#define   INTFORCE       *(volatile unsigned int *)(IRQMP_BASE+0x8)   //W   FIRST INT FORCE REGISTER
#define   INTCLEAR       *(volatile unsigned int *)(IRQMP_BASE+0xc)   //W   FIRST INT CLEAR REGISTER
#define   CTRL_RAM_BASE     0x80008000
#define   CTRL_REG_BASE     0x8000c000
#define   INT_MASK          0X0
#define   CFG_REG1          0x1
#define   CFG_REG2          0x2
#define   CFG_REG3          0x7
#define   CFG_REG4          0x8
#define   CFG_REG5          0x9
#define   START_RESET_REG   0x3
#define   MN_MN_POINTER       0x4
#define   INT_STATUS            0x6
#define   RT_STATUS             0xc
#define   RT_STATUS_RD        0xe
#define   RT_STACK_POINTER 0x3
#define control_word     0x180
#define   command_word ((0<<11)+(0<<10)+(5<<5)+0)
#define   command_word1   ((0<<11)+(1<<10)+(4<<5)+0)
#define   TR_BIT        (1<<10)
#define   RT_BUSY_BIT (1<<3)
#define   RT_ADDRESS   0xe //rt   address is 7
unsigned int rt_recv_pt,rt_send_pt;
unsigned int stack_data0[4]={0x0,0x0,0x320,0x108};
unsigned int stack_data[12]={0x0,0x0,0x320,0x108,
0x0,0x0,0x320,0x12e,
0x0,0x0,0x320,0x154,
};
unsigned int send_pt[32]=
{
0x0000,0xffff,0x0000,0xffff,
0x0000,0xffff,0x0000,0xffff,
0x0000,0xffff,0x0000,0xffff,
0x0000,0xffff,0x0000,0xffff,
```

```
0x0000,0xffff,0x0000,0xffff,
0x0000,0xffff,0x0000,0xffff,
0x0000,0xffff,0x0000,0xffff,
0x0000,0xffff,0x0000,0xffff
};
void M1553birq();
void bc_init();
void bc_send_message();
void bc_recv_message();
void delay();
unsigned int send_end_flag,send_num;
unsigned int rt_address_count,sub_address_count,data_count,data_count1;
```

MAIN.C

```
#include"bc.h"
main()
{
 printf("/************************************/");

 printf("\n       bc 控制器 BC 模式测试！");

 printf("\n       BC->RT0    SA5    MC32,RT0 SA4    MC 32");

 printf("\n/************************************/");
 INTMASK=0;
 INTCLEAR=(1<<10);
 catch_interrupt(M1553birq,10);
 delay();
 send_end_flag=1;

 bc_init();
 INTMASK=(1<<10);
 delay();
 *(volatile unsigned int *)(CTRL_REG_BASE+START_RESET_REG*4)=0x02;
 delay();
 while(send_end_flag)
 {
    M1553birq();

 }

////////////////////////////////////////////////////////////////////////////////
```

```
void M1553birq()
{
 unsigned int i,intsta,pending,message_block_address,message_block_address2,temp1[32],temp2[32],temp;
 //       printf("\n    coming ");
 pending=INTPEND;
 //              temp=0x5555;

 if(pending&1<<10)
 {

    intsta=*(volatile unsigned int *)(CTRL_REG_BASE + INT_STATUS*4);
    *(volatile unsigned int *)( CTRL_REG_BASE + START_RESET_REG*4 )=0x04;
    INTCLEAR = ( 1<<10 );
    for(i=0;i<4;i++);
    bc_send_message();
    delay();
    *(volatile unsigned int *)(CTRL_REG_BASE+START_RESET_REG*4)=0x02;
    delay();
    if(*(volatile unsigned int *)(0x80008420+34*4)!=*(volatile unsigned int *)(0x80008420+33*4))
       printf("\n last data word looped back is wrong!last data word looped back is %x actually!",*(volatile unsigned int *)(0x80008420+33*4));
    if(((*(volatile unsigned int *)(0x80008420+35*4)>>11)&0x1f)!=((*(volatile unsigned int *)(0x80008424)>>11)&0x1f))
       printf("\n status received is wrong !status received value is %x",*(volatile unsigned int *)(0x80008420+35*4));
    for(i=0;i<32;i++)
    {
       temp1[i] = *(volatile unsigned int *)(0x80008420+(i+2)*4);
    }
    if(*(volatile unsigned int*)0x80008000!=0x8000)
       printf("\n 块状态字有错，实际的块状态字是 %x ",*(volatile unsigned int*)0x80008000);
    delay();
    if(*(volatile unsigned int *)(0x800084bc)!=*(volatile unsigned int *)(0x800084c0))
       printf("\n transmit word looped back is wrong !transmit word value is %x actually!",*(volatile unsigned int *)(0x80008428));
    if(((*(volatile unsigned int *)(0x800084bc)>>11)&0x1f)!=((*(volatile unsigned int *)(0x800084c0)>>11)&0x1f))
       printf("\n status received is wrong!status received value is %x actually!",*(volatile unsigned int *)(0x80008428));
    for(i=0;i<32;i++)
    {
       temp2[i]=*(volatile unsigned int *)(0x800084b8+(i+4)*4);
    }
    for(i=0;i<32;i++)
    {
       if(temp1[i]!=temp2[i])
          printf("\n 期望收到的数是%x,实际收到的数是%x",temp1[i],temp2[i]);
    }
```

```
        if(*(volatile unsigned int*)0x80008010!=0x8010)
            printf("\n 块状态字有错，实际的块状态字是 %x ",*(volatile unsigned int*)0x80008010);
    }
    }


    void   bc_init()
    {
     unsigned int i,word_count,send_data, control_word_init,command_word_init;
      send_data=0x5555;

    //configure register

     *(volatile unsigned int *)(CTRL_REG_BASE+START_RESET_REG*4) = 0x1; //复位 1553B 寄存器
     *(volatile unsigned int *)(CTRL_REG_BASE+CFG_REG1*4) = 0x38;     //设置 BC 模式，时间标签使能，重试使能，
重试两次
     *(volatile unsigned int *)(CTRL_REG_BASE+CFG_REG2*4) = 0x388; // --time tag resolution 2 us
     *(volatile unsigned int *)(CTRL_REG_BASE+CFG_REG3*4) = 0x000; // --no override mode T/R* error
     *(volatile unsigned int *)(CTRL_REG_BASE+CFG_REG4*4) = 0x180; // --1ST retry ALT bus,2ND retry ALT bus
     *(volatile unsigned int *)(CTRL_REG_BASE+CFG_REG5*4) = 0x000; //response timeout select=00
     *(volatile unsigned int *)(CTRL_REG_BASE+INT_MASK*4) = 0x08;     //EOF 消息中断使能

     delay();

     for(i=0;i<8;i++)
     {
        *(volatile unsigned int *)(CTRL_RAM_BASE + i*4) = stack_data[i];//配置 RAM 堆栈
     }

     *(volatile unsigned int *)(CTRL_RAM_BASE + (0x100 + 0) *4)=0x0;//program the stack pointer

     *(volatile unsigned int *)(CTRL_RAM_BASE + (0x101 + 0) *4)=0xfffd;   //发送消息长度是 0xffff 的反码


     *(volatile unsigned int *)(CTRL_RAM_BASE+ 0x108*4) = control_word;
     *(volatile unsigned int *)(CTRL_RAM_BASE+ 0x109*4) = command_word;
     for(i=0;i<4;i++);

    //transmit data

     for(i=2;i<34;i++)
     {
        *(volatile unsigned int *)(CTRL_RAM_BASE+( 0x108+i)*4)=send_data++;
```

```
 }
 delay();
 *(volatile unsigned int *)(CTRL_RAM_BASE+ 0x12e*4) = control_word;
 *(volatile unsigned int *)(CTRL_RAM_BASE+ 0x12f*4) = command_word1;
 for(i=0;i<4;i++);
}

void    bc_send_message()
{

 unsigned int i,word_count,send_data, control_word_init,command_word_init;
 unsigned int message_block_address,block_state,send_data2,message_block_address1;
 // delay();
 message_block_address = *(volatile unsigned int *)((CTRL_RAM_BASE + (4*0 + 3)*4))*4 + CTRL_RAM_BASE;
 // temp1=message_block_address;
 // printf("\n is %x",(volatile unsigned int *)message_block_address);
 delay();
 *(volatile unsigned int *)(message_block_address)=control_word;
 *(volatile unsigned int *)(message_block_address+4)=command_word;
 for(i=2;i<34;i++)
 {
    *(volatile unsigned int *)(message_block_address+i*4)=send_pt[i-2];
 }
 for(i=0;i<4;i++);
 message_block_address1 = *(volatile unsigned int *)((CTRL_RAM_BASE + (4*1 + 3)*4))*4 + CTRL_RAM_BASE;
 *(volatile unsigned int *)(message_block_address1)=control_word;
 *(volatile unsigned int *)(message_block_address1+4)=command_word1;
 for(i=0;i<4;i++);
}

void delay()
{
 unsigned int i;
 for(i=0;i<0xfff;i++);
```

7.2 附录二：RT示例程序

RT.H

```
#include<stdio.h>

#define IRQMP_BASE                0x80000200
#define    INTMASK       *(volatile unsigned int *)(IRQMP_BASE+0x40)    //R/W    FIRST INT MASK REGISTER
```

```
#define    INTPEND       *(volatile unsigned int *)(IRQMP_BASE+0x4)    //R    FIRST INT PEND REGISTER
#define    INTFORCE      *(volatile unsigned int *)(IRQMP_BASE+0x8)    //W    FIRST INT FORCE REGISTER
#define    INTCLEAR      *(volatile unsigned int *)(IRQMP_BASE+0xc)    //W    FIRST INT CLEAR REGISTER
#define    CTRL_RAM_BASE     0x80008000
#define    CTRL_REG_BASE     0x8000c000
#define    INT_MASK          0X0
#define    CFG_REG1          0x1
#define    CFG_REG2          0x2
#define    CFG_REG3          0x7
#define    CFG_REG4          0x8
#define    CFG_REG5          0x9
#define    START_RESET_REG   0x3
#define    MN_MN_POINTER      0x4
#define    INT_STATUS           0x6
#define    RT_STATUS            0xc
#define    RT_STATUS_RD        0xe
#define    RT_STACK_POINTER 0x3
#define RT_ADDRESS 0xe   //地址 7

unsigned int add_data[] = {
0x100,0x141,0x142,0x143,0x144,0x145,0x146,0x147,0x148,0x149,
0x14a,0x14b,0x14c,0x14d,0x14e,0x14f,0x150,0x151,0x152,0x153,
0x154,0x155,0x156,0x157,0x158,0x159,0x15a,0x15b,0x15c,0x15d,
0x15e,0x161,0x162,0x163,0x164,0x165,0x166,0x167,0x168,0x169,
0x16a,0x16b,0x16c,0x16d,0x16e,0x16f,0x170,0x171,0x172,0x173,
0x174,0x175,0x176,0x177,0x178,0x179,0x17a,0x17b,0x17c,0x17d,
0x17e,0x187
};
unsigned int dat_data[] = {
0x000,0x260,0x260,0x260,0x260,0x260,0x260,0x260,0x260,0x260,
0x260,0x260,0x260,0x260,0x260,0x260,0x260,0x260,0x260,0x260,
0x260,0x260,0x260,0x260,0x260,0x260,0x260,0x260,0x260,0x260,
0x260,0x260,0x260,0x260,0x260,0x260,0x260,0x260,0x260,0x260,
0x260,0x260,0x260,0x260,0x260,0x260,0x260,0x260,0x260,0x260,
0x260,0x260
};
unsigned int send_pt[32]=
{
0x0000,0xffff,0x0000,0xffff,
0x0000,0xffff,0x0000,0xffff,
0x0000,0xffff,0x0000,0xffff,
0x0000,0xffff,0x0000,0xffff,
0x0000,0xffff,0x0000,0xffff,
0x0000,0xffff,0x0000,0xffff,
```

```
    0x0000,0xffff,0x0000,0xffff,
    0x0000,0xffff,0x0000,0xffff
    };
    void rt_init();
    void M1553birq();
    void rt_recv_message();
    void rt_send_message();
    void delay();

    unsigned int rt_recv_num,rt_address,rt_recv,rt_right_recv,rt_error_recv,recv_end_flag,command_word,word_count,num_count;
    unsigned int send_end_flag,recv_end_flag,rt_address_count,sub_address_count,stack_offset,stack_offset1,count,temp1[32],count;
```

MAIN.C

```
    #include"rt.h"
    main()
    {
    printf("/************************************/");
    printf("\n       BC 控制器 RT 模式测试！ ");
    printf("\n    BC-->RT0    SA1 MC32 RT0 SA1 MC32-->BC");
    printf("\n/************************************/");
    INTMASK=0;
    INTCLEAR=(1<<10);
    catch_interrupt(M1553birq,10);
    delay();
    send_end_flag=1;
    rt_recv_num=0;
    rt_address_count=0;
    sub_address_count=1;
    num_count=1;
    count=0;
    INTMASK=(1<<10);
    rt_init();
    delay();
    while(send_end_flag);
    void M1553birq()
     {
    unsigned int i=0,intsta,pending,message_block_address,message_block_address2,send_dat=0x5555,ww,temp2[32];
    pending=INTPEND;
     if(pending&1<<10)
    {
    intsta=*(volatile unsigned int *)(CTRL_REG_BASE+INT_STATUS*4);
    *(volatile unsigned int *)(CTRL_REG_BASE+START_RESET_REG*4)=0x04;
```

```
INTCLEAR=(1<<10);
if(intsta&0x01)
{
rt_recv_num++;

if(rt_recv_num<123008)
{
rt_recv_message();
delay();//
for(i=0;i<count;i++)
{
if(temp1[i]!=*(volatile unsigned int *)(CTRL_RAM_BASE+(0x260+i)*4))
{
 printf("\n 期望收到的数是%x, 实际收到的数是%x",temp1[i],*(volatile unsigned int *)(CTRL_RAM_BASE+(0x260+i)*4));
}
 }
 if(*(volatile unsigned int *)0x80008000 !=0x8000)
{
 printf("\n RT 模式块状态字有错");
printf("\n 实际的块状态字是%x",*(volatile unsigned int *)0x80008000);
                }
}
}
 }
void  rt_init()
{
  unsigned int i,send_data;
  send_data=0x5555;
      //configure register
     *(volatile unsigned int *)(  CTRL_REG_BASE+START_RESET_REG*4)=0x1;    //复位 1553B 寄存器
      *(volatile unsigned int *)(  CTRL_REG_BASE+CFG_REG1*4)=0x8780;    //设置 RT 模式，时间标签使能，重试使能，重试两次
      *(volatile unsigned int *)(  CTRL_REG_BASE+CFG_REG2*4)  = 0x388; // --time tag resolution 2 us
      *(volatile unsigned int *)(CTRL_REG_BASE+  CFG_REG3*4)  = 0x000; // --no override mode T/R* error
      *(volatile unsigned int *)(CTRL_REG_BASE+  CFG_REG4*4)  = 0x0188; // --1ST retry ALT bus,2ND retry ALT bus
      *(volatile unsigned int *)(CTRL_REG_BASE+CFG_REG5*4)  = 0x640|0x0; //set RT ADDRESS 0,TIME OVERRIDE ANSWER 130us
      *(volatile unsigned int *)(  CTRL_REG_BASE+INT_MASK*4)=0x0001;    //EOM 消息中断使能
      delay();
      printf("\n 配置堆栈");
          for(i=0;i<62;i++)
          {
```

```
                    *(volatile unsigned int *)(CTRL_RAM_BASE + add_data[i]*4) = dat_data[i];//配置 RT 工作模式
                }
                for (i=0;i<32;i++)      //fill the receive buffer
 {
        *(volatile unsigned int *)(CTRL_RAM_BASE + (0x260 + i)*4)= send_data++;
 }
   printf("\n  初始化结束！ ");
 }
void    rt_recv_message()
{
 unsigned int i,k,block_state,data_address,temp,tr_bit,send_data;
 //get block    stack    message
   delay();
   stack_offset1=*(volatile unsigned int *)(CTRL_REG_BASE+RT_STACK_POINTER*4)-4;
   data_address=*(volatile unsigned int *)(CTRL_RAM_BASE+(stack_offset1+2)*4);
   command_word=*(volatile unsigned int *)(CTRL_RAM_BASE+(stack_offset1+3)*4);
   send_data=0x6666;
   count=0;
   word_count=command_word&0x1f;
   tr_bit=command_word>>10;
   sub_address_count=(command_word>>5)&0x1f;
   rt_address_count=(command_word>>11)&0x1f;
   if(word_count==0)
        word_count=32;
        if(word_count==32)
        num_count=32;
}
void delay()
{
 unsigned int i;
 for(i=0;i<0xfff;i++);
}
```

7.3 附录三：BM示例程序

BM.H

```
#include<stdio.h>

#define   INTMASK       *(volatile unsigned int *)(0x80000090)    //R/W    FIRST INT MASK REGISTER
#define   INTPEND       *(volatile unsigned int *)(0x80000094)    //R    FIRST INT PEND REGISTER
#define   INTFORCE      *(volatile unsigned int *)(0x80000098)    //W    FIRST INT FORCE REGISTER
#define   INTCLEAR      *(volatile unsigned int *)(0x8000009C)    //W    FIRST INT CLEAR REGISTER
```

```
#define    CTRL_RAM_BASE      0X80054000
#define    CTRL_REG_BASE      0x80050000



#define    INT_MASK           0X0
#define    CFG_REG1           0x1
#define    CFG_REG2           0x2
#define    CFG_REG3           0x7
#define    CFG_REG4           0x8
#define    CFG_REG5           0x9
#define    START_RESET_REG    0x3
#define    MN_MN_POINTER       0x4
#define    INT_STATUS            0x6
#define    RT_STATUS             0xc
#define    RT_STATUS_RD         0xe
#define    RT_STACK_POINTER 0x3
#define RT_ADDRESS 0xe   //地址 7
#define BM_ADDRESS_ENBLE 0x280


unsigned int add_data[] = {
                                0x100,0x141,0x142,0x143,0x144,0x145,0x146,0x147,0x148,0x149,
                                0x14a,0x14b,0x14c,0x14d,0x14e,0x14f,0x150,0x151,0x152,0x153,
                                0x154,0x155,0x156,0x157,0x158,0x159,0x15a,0x15b,0x15c,0x15d,
                                0x15e,0x161,0x162,0x163,0x164,0x165,0x166,0x167,0x168,0x169,
                                0x16a,0x16b,0x16c,0x16d,0x16e,0x16f,0x170,0x171,0x172,0x173,
                                0x174,0x175,0x176,0x177,0x178,0x179,0x17a,0x17b,0x17c,0x17d,
                                0x17e,0x187
                               };
unsigned int dat_data[] = {
                                0x000,0x260,0x260,0x260,0x260,0x260,0x260,0x260,0x260,0x260,
                                0x260,0x260,0x260,0x260,0x260,0x260,0x260,0x260,0x260,0x260,
                                0x260,0x260,0x260,0x260,0x260,0x260,0x260,0x260,0x260,0x260,
                                0x260,0x260,0x260,0x260,0x260,0x260,0x260,0x260,0x260,0x260,
                                0x260,0x260,0x260,0x260,0x260,0x260,0x260,0x260,0x260,0x260,
                                0x260,0x260
                               };
unsigned int send_pt[32]=
                                {
                                    0x0000,0xffff,0x0000,0xffff,
                                    0x0000,0xffff,0x0000,0xffff,
                                    0x0000,0xffff,0x0000,0xffff,
                                    0x0000,0xffff,0x0000,0xffff,
```

```
                                        0x0000,0xffff,0x0000,0xffff,
                                        0x0000,0xffff,0x0000,0xffff,
                                        0x0000,0xffff,0x0000,0xffff,
                                        0x0000,0xffff,0x0000,0xffff
                                    };


    void bm_init();
    void M1553birq();
    void bm_recv_message();
    void delay();

    unsigned                                                                                                    int
rt_recv_num,rt_address,rt_recv,rt_right_recv,rt_error_recv,recv_end_flag,command_word,word_count,num_count;
    unsigned int send_end_flag,recv_end_flag,rt_address_count,sub_address_count,stack_offset,stack_offset1,count;
```

MAIN.C

```
    #include"bm.h"
    main()
    {
            printf("/************************************/");
            printf("\n       BC 控制器 bm 模式测试！ ");
            printf("\n       BC-->RT0    ");
            printf("\n/************************************/");
            INTMASK=0;
            INTCLEAR=(1<<12);
            catch_interrupt(M1553birq,12);
            delay();
            send_end_flag=1;
            rt_recv_num=0;
            rt_address_count=0;
            sub_address_count=1;
            num_count=0;
            count=0;
            INTMASK=(1<<12);
            rt_init();
            *(volatile unsigned int *)(CTRL_REG_BASE+ START_RESET_REG *4)         = 0x2;
            while(send_end_flag);
    void M1553birq()
    {
        unsigned int i=0,intsta,pending,message_block_address,message_block_address2,send_dat=0x5555,ww;

        pending=INTPEND;
```

```
        intsta=*(volatile unsigned int *)(CTRL_REG_BASE+INT_STATUS*4);
        *(volatile unsigned int *)(CTRL_REG_BASE+START_RESET_REG*4)=0x06;
        INTCLEAR=(1<<12);
        bm_recv_message();
    }
    void bm_init()
    {
      unsigned int i,send_data;
      send_data=0x5555;
    //configure register
            *(volatile unsigned int *)(    CTRL_REG_BASE+START_RESET_REG*4)=0x1;    //复位 1553B 寄存器
     *(volatile unsigned int *)(    CTRL_REG_BASE+CFG_REG1*4)=0x4000;      //设置 bm 模式
     *(volatile unsigned int *)(    CTRL_REG_BASE+CFG_REG2*4)    = 0x388; // --time tag resolution 2 us
     *(volatile unsigned int *)(CTRL_REG_BASE+      CFG_REG3*4)  = 0x000; // --no override mode T/R* error
     *(volatile unsigned int *)(CTRL_REG_BASE+      CFG_REG4*4)  = 0x0188; // --1ST retry ALT bus,2ND retry ALT bus
    //      *(volatile unsigned int *)(CTRL_REG_BASE+CFG_REG5*4)       =  0x640|0x0;  //set  RT  ADDRESS  0,TIME
OVERRIDE  ANSWER  130us
     *(volatile unsigned int *)(    CTRL_REG_BASE+INT_MASK*4)=0x0001;      //EOM 消息中断使能
     delay();
     printf("\n 配置堆栈");
     *(volatile unsigned int *)(    CTRL_RAM_BASE+0x100*4)=0x0;

     for(i=0;i<0x80;i++)
     {

       *(volatile unsigned int *)(CTRL_RAM_BASE +( BM_ADDRESS_ENBLE + i)*4) = 0xffff;//配置 RT 工作模式
     }
     printf("\n bm 初始化结束！");
     for(i=0;i<0xff;i++);
     }
    void   bm_recv_message()
    {
     unsigned int i,k,block_state,data_address,temp,tr_bit;
    //get block   stack   message
     delay();
     stack_offset1=*(volatile unsigned int *)(CTRL_REG_BASE+RT_STACK_POINTER*4)-4;
     data_address=*(volatile unsigned int *)(CTRL_RAM_BASE+(stack_offset1+2)*4);
     command_word=*(volatile unsigned int *)(CTRL_RAM_BASE+(stack_offset1+3)*4);

     word_count=command_word&0x1f;
     tr_bit=command_word>>10;
     sub_address_count=(command_word>>5)&0x1f;
     rt_address_count=(command_word>>11)&0x1f;
```

```
//      printf("\n %x",*(volatile unsigned int *)0x80050004);
}
}
void delay()
{
unsigned int    i;
 for(i=0;i<0xfff;i++);
}
```

8. 参考资料

1. 数字式时分制指令/响应型多路传输数据总线 GJB 289A-97，中华人民共和国国家军用标准。
2. ACE/Mini-ACE Series BC/RT/MT Advanced Communication Engine Interated 1553 Terminal User's Guide .REVJ-3 MARCH.2003,Data Device Corporation.
3. MIL-STD-1553 DESINER'S GUIDE SIXTH EDITION ,Data Device Corporation.